

RRRRRRRR	SSSSSSSS	DDDDDDDD	EEEEEEEEE	RRRRRRRR	MM	MM
RRRRRRRR	SSSSSSSS	DDDDDDDD	EEEEEEEEE	RRRRRRRR	MM	MM
RR RR	SS DD	DD DD	EE EE	RR RR	MMMM	MMMM
RR RR	SS DD	DD DD	EE EE	RR RR	MMMM	MMMM
RR RR	SS DD	DD DD	EE EE	RR RR	MM MM	MM MM
RR RR	SS DD	DD DD	EE EE	RR RR	MM MM	MM MM
RRRRRRRR	SSSSSS	DD DD	EEEEEEEEE	RRRRRRRR	MM	MM
RRRRRRRR	SSSSSS	DD DD	EEEEEEEEE	RRRRRRRR	MM	MM
RR RR	SS DD	DD DD	EE EE	RR RR	MM	MM
RR RR	SS DD	DD DD	EE EE	RR RR	MM	MM
RR RR	SS DD	DD DD	EE EE	RR RR	MM	MM
RR RR	SS DD	DD DD	EE EE	RR RR	MM	MM
RR RR	SSSSSSSS	DDDDDDDD	EEEEEEEEE	RR RR	MM	MM
RR RR	SSSSSSSS	DDDDDDDD	EEEEEEEEE	RR RR	MM	MM

LL	IIIIII	SSSSSSSS
LL	IIIIII	SSSSSSSS
LL	II	SS
LLLLLLLL	IIIIII	SSSSSSSS
LLLLLLLL	IIIIII	SSSSSSSS

```
1 0001 0 XTITLE 'Misc. error checks and messages.'  
2 0002 0 MODULE RSDERM (   
3 0003 0 IDENT = 'V04-000'  
4 0004 0 XBLISS32[  
5 0005 0 ADDRESSING_MODE(INTERNAL=LONG_RELATIVE,NONEXTERNAL=LONG_RELATIVE)  
6 0006 0 ]  
7 0007 0 ) =  
8 0008 1 BEGIN  
9 0009 1 *****  
10 0010 1 *  
11 0011 1 *  
12 0012 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY  
13 0013 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.  
14 0014 1 * ALL RIGHTS RESERVED.  
15 0015 1 *  
16 0016 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  
17 0017 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE  
18 0018 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER  
19 0019 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  
20 0020 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY  
21 0021 1 * TRANSFERRED.  
22 0022 1 *  
23 0023 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE  
24 0024 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT  
25 0025 1 * CORPORATION.  
26 0026 1 *  
27 0027 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS  
28 0028 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.  
29 0029 1 *  
30 0030 1 *  
31 0031 1 *****  
32 0032 1 *  
33 0033 1 **  
34 0034 1 FACILITY: DSR (Digital Standard RUNOFF) / DSRPLUS  
35 0035 1  
36 0036 1 ABSTRACT: Residual error messages. Issues error messages for various  
37 0037 1 error conditions for which no error messages have been  
38 0038 1 issued.  
39 0039 1  
40 0040 1 Also contains error detection and error message routines  
41 0041 1 that are called from more than one location.  
42 0042 1  
43 0043 1 ENVIRONMENT: Transportable  
44 0044 1  
45 0045 1 AUTHOR: R.W.Friday CREATION DATE: June, 1978  
46 0046 1
```

48 0047 1 %SBTTL 'Revision History'
49 0048 1
50 0049 1 MODIFIED BY:
51 0050 1
52 0051 1 006 REM00006 Ray Marshall 22-June-1983
53 0052 1 Modified routine TSTTFF to now look to see if we are within a
54 0053 1 .LITERAL. If so, it issues an error and closes forces a
55 0054 1 .END LITERAL.
56 0055 1
57 0056 1 005 KFA00005 Ken Alden 18-Mar-1983
58 0057 1 Made SAVE/RESTORE related items visible to DSR.
59 0058 1
60 0059 1 004 REM00004 Ray Marshall 07-Mar-1983
61 0060 1 Global edit of all modules. Updated module names, idents,
62 0061 1 copyright dates. Changed require files to BLISS library.
63 0062 1
64 0063 1 !--

```
66 0064 1 %SBTTL 'Module Level Declarations'  
67 0065 1  
68 0066 1 REQUIRE 'REQ:RNODEF';  
69 0197 1  
70 0198 1  
71 0199 1 | TABLE OF CONTENTS:  
72 0200 1  
73 0201 1 FORWARD ROUTINE  
74 0202 1 tstblk : NOVALUE,  
75 0203 1 tstdnd : NOVALUE,  
76 0204 1 tstdres : NOVALUE,  
77 0205 1 tstdtfe : NOVALUE,  
78 0206 1 pntbac : NOVALUE,  
79 0207 1 negind : NOVALUE,  
80 0208 1 remneg : NOVALUE,  
81 0209 1 xmarg : NOVALUE,  
82 0210 1 remmrg : NOVALUE;  
83 0211 1  
84 0212 1  
85 0213 1 | INCLUDE FILES:  
86 0214 1 |  
87 0215 1  
88 0216 1 LIBRARY 'NXPORT:XPORT';  
89 0217 1  
90 0218 1 %IF DSRPLUS %THEN  
91 0219 1 LIBRARY 'REQ:DPLLIB';  
92 0220 1 ! DSRPLUS BLISS Library  
93 0221 1 %ELSE  
94 0222 1 LIBRARY 'REQ:DSRLIB';  
95 0223 1 ! DSR BLISS Library  
96 0224 1  
97 0225 1 | MACROS:  
98 0226 1  
99 0227 1  
100 0228 1 | EQUATED SYMBOLS:  
101 0229 1  
102 0230 1  
103 0231 1 | OWN STORAGE:  
104 0232 1  
105 0233 1  
106 0234 1 | EXTERNAL REFERENCES:  
107 0235 1  
108 0236 1  
109 0237 1 EXTERNAL  
110 0238 1 gca : gca_definition,  
111 0239 1 ifstk : ifstack,  
112 0240 1 savstk : savstack,  
113 0241 1 irac : irac_definition;  
114 0242 1  
115 0243 1 EXTERNAL  
116 0244 1 frmstd,  
117 0245 1 frmstk : form_stack;  
118 0246 1  
119 0247 1 EXTERNAL LITERAL  
120 0248 1 rnfbak,  
121 0249 1 rnfldc,  
122 0250 1 rnfmei;
```

!Error messages

123 0251 1 rnmrc,
124 0252 1 rnmrg,
125 0253 1 rnmrs,
126 0254 1 rnmnia,
127 0255 1 rnmnic,
128 0256 1 rnmfkc,
129 0257 1 rnmftfe;
130 0258 1
131 0259 1 EXTERNAL ROUTINE
132 0260 1 erm,
133 0261 1 erm1,
134 0262 1 erm2,
135 0263 1 lit,
136 0264 1 savres,
137 0265 1 stkfrm;
138 0266 1
139 0267 1

```
: 141 0268 1 %SBTTL 'TSTBLK -- test for unclosed .NOTE or .LIST directives'  
142 0269 1 GLOBAL ROUTINE tstblk (depth) : NOVALUE =  
143 0270 1  
144 0271 1 |++  
145 0272 1 | FUNCTIONAL DESCRIPTION  
146 0273 1  
147 0274 1 | This routine checks to see if FRMSTK is ok: i.e., certain open  
148 0275 1 | .NOTE and .LIST commands have been closed. If there are any  
149 0276 1 | unclosed commands it issues an error message and pops the stack.  
150 0277 1  
151 0278 1 | FORMAL PARAMETERS:  
152 0279 1  
153 0280 1 | DEPTH specifies the maximum allowed depth of .REQUIRE commands.  
154 0281 1  
155 0282 1 | IMPLICIT INPUTS: None  
156 0283 1  
157 0284 1 | IMPLICIT OUTPUTS: None  
158 0285 1  
159 0286 1 | ROUTINE VALUE:  
160 0287 1 | COMPLETION CODES: None  
161 0288 1  
162 0289 1 | SIDE EFFECTS: None  
163 0290 1  
164 0291 1 |--  
165 0292 1  
166 0293 2 | BEGIN  
167 0294 2  
168 0295 2 | IF .frmstd EQL 0 THEN  
169 0296 2 | | RETURN; !No unclosed lists, notes, or literals.  
170 0297 2  
171 0298 2 | IF .depth GTR .frmstk [.frmstd, frmstk_req_d] THEN  
172 0299 2 | | RETURN; !Unclosed lists or notes, but in files still open.  
173 0300 2  
174 0301 2 | erm (rnftfe, 0, 0); !There are unclosed lists/notes to be reported.  
175 0302 2 | !Now point the user back to those commands that have not been closed  
176 0303 2  
177 0304 2 | WHILE ((.frmstd GTR 0) AND (.depth LEQ .frmstk [.frmstd, frmstk_req_d])) DO  
178 0305 3 | | BEGIN  
179 0306 3 | | | pntbac ();  
180 0307 3 | | | stkfrm (-i); !Unstack one entry, regardless of identity.  
181 0308 2 | | | END;  
182 0309 2  
183 0310 1 | | END; !End of TSTBLK
```

```
.TITLE RSDERM Misc. error checks and messages.  
.IDENT \V04-000\  
.EXTRN GCA, IFSTK, SAVSTK  
.EXTRN IRAC, FRMSID, FRMSTK  
.EXTRN RNFBAK, RNFLDE, RNFMEI  
.EXTRN RNFMRG, RNFMRG, RNFMRG  
.EXTRN RNFNIA, RNFNIC, RNFSC  
.EXTRN RNFTFE, ERM, EML  
.EXTRN ERMN, LIT, SAVRES  
.EXTRN STKFRM
```

```

.PSECT $CODE$,NOWRT,2

      000C 00000
      EF 9E 00002
      EF 9E 00009
      63 D0 00010
      3B 13 00013
      OF C4 00015
      AC D1 00018
      31 14 0001D
      7E 7C 0001F
      8F DD 00021
      03 FB 00027
      63 D0 0002E 1$:
      10 15 00031
      OF C4 00033
      AC D1 00036
      13 14 00038
      00 FB 0003D
      01 CE 00044
      01 FB 00047
      DE 11 0004E
      04 00050 2$:
      .ENTRY TSTBLK, Save R2,R3 : 0269
      MOVAB FRMSTD, R3
      MOVAB FRMSTK-20, R2
      MOVL FRMSTD, R0
      BEQL 2$ : 0295
      MULL2 #15, R0
      CMPL DEPTH, FRMSTK-20[R0]
      BGTR 2$ : 0298
      CLRQ -(SP)
      PUSHL #RNFTFE
      CALLS #3, ERM
      MOVL FRMSTD, R0
      BLEQ 2$ : 0301
      MULL2 #15, R0
      CMPL DEPTH, FRMSTK-20[R0]
      BGTR 2$ : 0304
      CALLS #0, PNTBAC
      MNEGL #1, -(SP) : 0306
      CALLS #1, STKFRM : 0307
      BRB 1$ : 0304
      RET : 0310

```

: Routine Size: 81 bytes. Routine Base: \$CODE\$ + 0000

```

185 0311 1 %SBTTL 'TSTCND -- Check and report missing .ENDIFs'
186 0312 1 GLOBAL ROUTINE tscnd (depth) : NOVALUE =
187 0313 1
188 0314 1 ++
189 0315 1 FUNCTIONAL DESCRIPTION
190 0316 1
191 0317 1 FORMAL PARAMETERS:
192 0318 1
193 0319 1 DEPTH specifies the maximum allowed depth of .REQUIRE commands.
194 0320 1
195 0321 1 IMPLICIT INPUTS: None
196 0322 1
197 0323 1 IMPLICIT OUTPUTS: None
198 0324 1
199 0325 1 ROUTINE VALUE:
200 0326 1 COMPLETION CODES: None
201 0327 1
202 0328 1
203 0329 1 SIDE EFFECTS: None
204 0330 1
205 0331 1 --
206 0332 1
207 0333 2 BEGIN
208 0334 2 !Check for missing .ENDIF commands. This is temporary coding. In a
209 0335 2 !later version this code should point back to the opening commands,
210 0336 2 !like the code above does for .LIST and .NOTE commands.
211 0337 2
212 0338 2 IF .ifstk [0, ifstk_depth] EQL 0 THEN
213 0339 2 RETURN: !There are no missing .ENDIF commands.
214 0340 2
215 0341 2 IF .depth EQL 0 THEN
216 0342 2 erml (rnfm ei) ! Missing .ENDIF commands, and end of file.
217 0343 2 ELSE
218 0344 2 IF .ifstk [.ifstk [0, ifstk_depth], ifstk_req_d] GEQ .depth THEN
219 0345 2 erml (rnfskc); ! Missing .ENDIF commands detected when
220 0346 2 ! unstacking .REQUIRE files
221 0347 2
222 0348 1 END: !End of TSTCND

```

50 0000000G	EF	0000 0000	.ENTRY	TSTCND. Save nothing	0312
		00002	MOVL	IFSTK, R0	0338
	2A	13 0009	BEQL	3\$	0341
04	AC	D5 000B	TSTL	DEPTH	0342
		08 12 000E	BNEQ	1\$	0344
000000UOG	8F	DD 00010	PUSHL	#RNFM EI	0345
		16 11 00016	BRB	2\$	0348
50	20	C4 00018	MULL2	#32, R0	
04	AC	0000000GEF	PUSHAB	IFSTK+16[R0]	
		40 9F 0001B	CMPL	@(SP)+, DEPTH	
	9E	D1 00022	BLSS	3\$	
0000000G	0D	19 00026	PUSHL	#RNFSKC	
EF	8F	DD 00028	CALLS	#1, ERML	
	01	FB 0002E			
0000000G	04	00035	3\$:	RET	

RSDERM
V04-000

Misc. error checks and messages.
TSTCND -- Check and report missing .ENDIFs

L 11
16-Sep-1984 01:40:15
14-Sep-1984 13:07:56

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]RSDERM.BLI;1

Page 8
(5)

; Routine Size: 54 bytes, Routine Base: \$CODE\$ + 0051

```
224 0349 1 %SBTTL 'TSTRES -- Check for, report, and restore missing .RESTORES'  
225 0350 1 ROUTINE tstres (depth) : NOVALUE =  
226 0351 1  
227 0352 1 ++  
228 0353 1 FUNCTIONAL DESCRIPTION  
229 0354 1  
230 0355 1 This routine checks to see if SAVSTK is ok: i.e., certain open  
231 0356 1 .SAVE/SAVE ALL and .RESTORE commands have been closed. If there are any  
232 0357 1 unclosed commands it issues an error message and pops the stack.  
233 0358 1  
234 0359 1 FORMAL PARAMETERS:  
235 0360 1  
236 0361 1 DEPTH specifies the maximum allowed depth of .REQUIRE commands.  
237 0362 1  
238 0363 1 IMPLICIT INPUTS: None  
239 0364 1  
240 0365 1 IMPLICIT OUTPUTS: None  
241 0366 1  
242 0367 1 ROUTINE VALUE:  
243 0368 1 COMPLETION CODES: None  
244 0369 1  
245 0370 1 SIDE EFFECTS: None  
246 0371 1  
247 0372 1 !--  
248 0373 1  
249 0374 2 BEGIN  
250 0375 2 OWN  
251 0376 2 x;  
252 0377 2  
253 0378 2 LOCAL  
254 0379 2 hold_iseqn,  
255 0380 2 hold_page;  
256 0381 2  
257 0382 2 x = .savstk [0, savstk_depth];  
258 0383 2  
259 0384 2 IF (.x EQL 0) THEN  
260 0385 2 RETURN; ! No unclosed .SAVEs.  
261 0386 2  
262 0387 2 IF .depth GTR .savstk [.x, savstk_req_d] THEN  
263 0388 2 RETURN; ! Unclosed SAVEs but in files still open.  
264 0389 2  
265 0390 2 !There are unclosed saves to be reported.  
266 0391 2  
267 0392 2 erml (rnfmrs); ! Missing .ReStore commands, at end of file.  
268 0393 2  
269 0394 2 ! Now point the user back to the point where the corresponding  
270 0395 2 ! SAVE/SAVEALL was made.  
271 0396 2  
272 0397 4 WHILE ( (.savstk [0, savstk_depth] GTR 0)  
273 0398 3 AND  
274 0399 2 (.depth LEQ .savstk [.savstk [0, savstk_depth], savstk_req_d]) ) DO  
275 0400 3 BEGIN  
276 0401 3 hold_page = .irac_ipagen;  
277 0402 3 hold_iseqn = .irac_iseqn;  
278 0403 3 irac_ipagen = .savstk [.savstk [0, savstk_depth], savstk_ipagen];  
279 0404 3 irac_iseqn = .savstk [.savstk [0, savstk_depth], savstk_iseqn];  
280 0405 3
```

```

281      0406 3      IF .gca_req_depth NEQ 0 THEN | Still in a require file, rewind
282          0407 3      erm (rnfbak | stack. Otherwise just quit.
283          0408 3      ..savstk [.savstk [0, savstk_depth], savstk_fspecp]
284          0409 3      ..savstk [.savstk [0, savstk_depth], savstk_fspecc] );
285          0410 3
286          0411 3      irac_ipagen = .hold_page;
287          0412 3      irac_iseqn = .hold_iseqn;
288          0413 3      savres (h_restore,-1);   ! Unstack one entry, regardless of identity.
289          0414 2      END;
290          0415 2
291          0416 1      END;           ! End of TSTRS

```

```

.PSECT $0WN$,NOEXE,2
00000 X: .BLKB 4

```

```
.PSECT $CODE$,NOWRT,2
```

56 00000000'	EF 9E 00002	007C 00000 TSTRS: .WORD Save R2,R3,R4,R5,R6 : 0350
55 00000000G	EF 9E 00009	MOVAB X, R6
54 00000000G	EF 9E 00010	MOVAB IRAC+12, R5
66 50	64 D0 00017	MOVAB SAVSTK, R4
	66 D0 0001A	MOVL SAVSTK, X
	7F 13 0001D	MOVL X, R0
50	1C C4 0001F	BEQL 3\$
50	OC A440 9F 00022	MULL2 #28, R0
9E	04 AC D1 00026	PUSHAB SAVSTK+12[R0]
	72 14 0002A	CMPL DEPTH, @(SP)+
	00000000G 8F DD 0002C	BGTR 3\$
00000000G EF	01 FB 00032	PUSHL #RNFMRS
51	64 D0 00039	CALLS #1, ERML
	60 15 0003C	1\$: MOVL SAVSTK, R1 : 0397
50	51 1C C5 0003E	BLEQ 3\$
	OC A440 9F 00042	MULL3 #28, R1, R0
9E	04 AC D1 00046	PUSHAB SAVSTK+12[R0]
	52 14 0004A	CMPL DEPTH, @(SP)+
	52 65 D0 0004C	BGTR 3\$
50	53 FC A5 D0 0004F	MOVL IRAC+12, HOLD PAGE
	51 1C C5 00053	MOVL IRAC+8, HOLD_ISEQN : 0402
	10 A440 9F 00057	MULL3 #28, R1, R0 : 0403
65	9E D0 0005B	PUSHAB SAVSTK+16[R0]
	14 A440 9F 0005E	MOVL @(SP)+, IRAC+12
FC A5	9E D0 00062	PUSHAB SAVSTK+20[R0] : 0404
	00000000G EF D5 00066	MOVL @(SP)+, IRAC+8
	19 13 0006C	TSTL GCA+18\$
	1C A440 9F 0006E	BEQL 2\$
	9E DD 00072	PUSHAB SAVSTK+28[R0] : 0409
	18 A440 9F 00074	PUSHL @(SP)+
	9E DD 00078	PUSHAB SAVSTK+24[R0] : 0408
00000000G EF	8F DD 0007A	PUSHL @(SP)+
65	03 FB 00080	CALLS #3, ERM : 0407
	52 D0 00087	MOVL HOLD_PAGE, IRAC+12 : 0411
2\$:		

RSDERM
V04-000

Misc. error checks and messages.
TSTRS -- Check for, report, and restore missin

B 12
16-Sep-1984 01:40:15
14-Sep-1984 13:07:56

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]RSDERM.BLI;1

Page 11
(6)

RSK
V04

FC	A5	53	D0 0008A	MOVL	HOLD_ISEQN, IRAC+8	:	0412
7E		01	CE 0008E	MNEG.L	#1 -(SP)	:	0413
7E	AF	8F	9A 00091	MOVZBL	#175, -(SP)	:	
00000000G	EF	02	FB 00095	CALLS	#2, SAVRES	:	0397
		9B	11 0009C	BRB	1\$:	
		04	0009E 3\$:	RET		:	0416

: Routine Size: 159 bytes, Routine Base: \$CODE\$ + 0087

```

: 293 0417 1 GLOBAL ROUTINE tstffe (depth) : NOVALUE =
: 294 0418 1
: 295 0419 1 !++
: 296 0420 1 FUNCTIONAL DESCRIPTION
: 297 0421 1
: 298 0422 1 This routine does nothing more than merely pass control to each of
: 299 0423 1 the respective test routines to check for unclosed lists, saves, etc.
: 300 0424 1 Oh, one exception: It now checks for being within a .LITERAL. If it
: 301 0425 1 finds this to be true, it issues an error and closes the literal.
: 302 0426 1
: 303 0427 1 FORMAL PARAMETERS:
: 304 0428 1
: 305 0429 1 DEPTH specifies the maximum allowed depth of .REQUIRE commands.
: 306 0430 1
: 307 0431 1 IMPLICIT INPUTS: None
: 308 0432 1
: 309 0433 1 IMPLICIT OUTPUTS: None
: 310 0434 1
: 311 0435 1 ROUTINE VALUE:
: 312 0436 1 COMPLETION CODES: None
: 313 0437 1
: 314 0438 1 SIDE EFFECTS: None
: 315 0439 1
: 316 0440 1 !--
: 317 0441 1
: 318 0442 2 BEGIN
: 319 0443 2
: 320 0444 2 IF .gca_literal THEN ! If within a literal;
: 321 0445 3 BEGIN
: 322 0446 3 erm (rnflde ! say that it wasn't closed,
: 323 0447 3 ,CH$PTR(UPLIT('At or near end of document'))! and where
: 324 0448 3 ,26);
: 325 0449 3 lit( h_end_literal ) ! and then close it.
: 326 0450 2 END;
: 327 0451 2 tstblk (.depth); ! Check for unclosed lists, notes, and literals.
: 328 0452 2 tscnd (.depth); ! Check for unclosed ifs and ifnots.
: 329 0453 2 tstres (.depth) ! Check for unclosed saves.
: 330 0454 2
: 331 0455 1 END: ! End of TSTTFF

```

```

20 64 6E 65 20 72 61 65 6E 20 72 6F 20 74 41 00000 P.AAA: .ASCII \At or near end of document\<0><0>
00 00 74 6E 65 6D 75 63 6F 64 20 66 6F 0000F

```

```
.PSECT $PLITS,NOWRT,NOEXE,2
```

52	FED4	0004 00000	.ENTRY	TSTTFF, Save R2	0417
1E	00000000G	CF 9E 00002	MOVAB	TSTBLK, R2	0444
		EF E9 00007	BLBC	GCA+80, 1\$	0446
		1A DD 0000E	PUSHL	#26	0447
	00000000'	EF 9F 00010	PUSHAB	P.AAA	0446
	00000000G	8F DD 00016	PUSHL	#RNFLDE	S

```
.PSECT $CODE$,NOWRT,2
```

00000000G	EF	03	FB 0001C	CALLS	#3 ERM
00000000G	EF	01	FB 00023	PUSHL	#60
		04	AC DD 0002C	CALLS	#1 LIT
	62	01	FB 0002F	PUSHL	DEPTH
		04	AC DD 00032	CALLS	#1, TSTBLK
	51 A2	01	FB 00035	PUSHL	DEPTH
		04	AC DD 00039	CALLS	#1, TSTCND
0087	C2	01	FB 0003C	PUSHL	DEPTH
		04	00041	CALLS	#1, TSTRES
				RET	

: L
: L
: M
: C

; Routine Size: 66 bytes, Routine Base: \$CODE\$ + 0126

: 332 0456 1

```

: 334
: 335 0457 1 GLOBAL ROUTINE pntbac : NOVALUE =
: 336 0458 1 ++
: 337 0459 1 : FUNCTIONAL DESCRIPTION
: 338 0460 1 :
: 339 0461 1 : This routine saves the input file line/page information. It then
: 340 0462 1 : substitutes information from FRMSTK for that information. Then ERM is
: 341 0463 1 : called to output a message indicating where to look in the input file
: 342 0464 1 : for the command that made the stack entry. But ERM does not have
: 343 0465 1 : enough flexibility to accept all that information as parameters.
: 344 0466 1 : However, it can pick it up from IRAC, which it does. In other words,
: 345 0467 1 : this routine fakes out ERM a bit.
: 346 0468 1 :
: 347 0469 1 : After ERM returns, the original information is restored.
: 348 0470 1 :
: 349 0471 1 :-- BEGIN
: 350 0472 2
: 351 0473 2 : LOCAL
: 352 0474 2   hold_iseqn,
: 353 0475 2   hold_page;
: 354 0476 2
: 355 0477 2   hold_page = .irac_ipagen;
: 356 0478 2   hold_iseqn = .irac_iseqn;
: 357 0479 2   irac_ipagen = .frmstk [.frmstd, frmstk_ipagen];
: 358 0480 2   irac_iseqn = .frmstk [.frmstd, frmstk_iseqn];
: 359 0481 2   erm Trnfbak
: 360 0482 2     ..frmstk [.frmstd, frmstk_fspecp]
: 361 0483 2     ..frmstk [.frmstd, frmstk_fspecc] );
: 362 0484 2   irac_ipagen = .hold_page;
: 363 0485 2   irac_iseqn = .hold_iseqn;
: 364 0486 2
: 0487 1   END;                                !End of PNTBAC

```

		003C 00000	.ENTRY	PNTBAC, Save R2,R3,R4,R5	: 0457
	55 00000000G	FF 9E 00002	MOVAB	FRMSTK-32, R5	
	54 00000000G	EF 9E 00009	MOVAB	IRAC+12, R4	: 0478
	52	64 D0 00010	MOVL	IRAC+12, HOLD PAGE	
	53	A4 D0 00013	MOVL	IRAC+8, HOLD_ISEQN	: 0479
	EF	0F C5 00017	MULL3	#15, FRMSTD -R0	: 0480
	64	6540 D0 0001F	MOVL	FRMSTK-32[R0], IRAC+12	
50	FC A4	FC A540 D0 00023	MOVL	FRMSTK-36[R0], IRAC+8	: 0481
		08 A540 DD 00029	PUSHL	FRMSTK-24[R0]	: 0484
		04 A540 DD 0002D	PUSHL	FRMSTK-28[R0]	: 0483
	00000000G	8F DD 00031	PUSHL	#RNFBAK	: 0482
	EF	03 FB 00037	CALLS	#3, ERM	
	64	52 D0 0003E	MOVL	HOLD_PAGE, IRAC+12	: 0485
	FC A4	53 D0 00041	MOVL	HOLD_ISEQN, IRAC+8	: 0486
		04 00045	RET		: 0487

: Routine Size: 70 bytes. Routine Base: \$CODES + 0168

: 365 0488 1

```

: 367 0489 1 GLOBAL ROUTINE negind : NOVALUE =
: 368 0490 1
: 369 0491 1 ++
: 370 0492 1 FUNCTIONAL DESCRIPTION:
: 371 0493 1
: 372 0494 1 NEGIND keeps track of attempted indents past the left side of the page.
: 373 0495 1 To avoid issuing so many error messages that the user would be annoyed,
: 374 0496 1 it only issues an error message for the first such attempt. Other
: 375 0497 1 violations are simply counted; the summary count is output by REMNEG
: 376 0498 1 later.
: 377 0499 1
: 378 0500 1 FORMAL PARAMETERS: None
: 379 0501 1
: 380 0502 1 IMPLICIT INPUTS None
: 381 0503 1
: 382 0504 1 IMPLICIT OUTPUTS None
: 383 0505 1
: 384 0506 1 ROUTINE VALUE:
: 385 0507 1 COMPLETION CODES: None
: 386 0508 1
: 387 0509 1 SIDE EFFECTS: None
: 388 0510 1
: 389 0511 1 --
: 390 0512 1
: 391 0513 2 BEGIN
: 392 0514 2
: 393 0515 2 IF .gca_nia EQL 0 THEN
: 394 0516 2   erml(rnfnia);
: 395 0517 2
: 396 0518 2   gca_nia = .gca_nia + 1;
: 397 0519 1 END;                                !End of NEGIND

```

52 00000000G	EF 0004 00000	.ENTRY NEGIND, Save R2	: 0489
	9E 00002	MOVAB GCA+160, R2	: 0515
	62 D5 00009	TSTL GCA+160	: 0516
00000000G	0D 12 0000B	BNEQ 1\$: 0518
00000000G EF	8F DD 0000D	PUSHL #RNFNIA	: 0519
	01 FB 00013	CALLS #1, ERML	
	62 D6 0001A 1\$:	INCL GCA+160	
	04 0001C	RET	

; Routine Size: 29 bytes. Routine Base: \$CODE\$ + 01AE

; 398 0520 1

```

: 400 0521 1 GLOBAL ROUTINE remneg : NOVALUE =
: 401 0522 1
: 402 0523 1 ++
: 403 0524 1 FUNCTIONAL DESCRIPTION:
: 404 0525 1
: 405 0526 1 Issues an error message indicating how many negative
: 406 0527 1 indents have been detected. Then resets the count to zero.
: 407 0528 1
: 408 0529 1 Used together with NEGIND.
: 409 0530 1
: 410 0531 1 FORMAL PARAMETERS: None
: 411 0532 1
: 412 0533 1 IMPLICIT INPUTS: None
: 413 0534 1
: 414 0535 1 IMPLICIT OUTPUTS: None
: 415 0536 1
: 416 0537 1 ROUTINE VALUE:
: 417 0538 1 COMPLETION CODES: None
: 418 0539 1
: 419 0540 1 SIDE EFFECTS: None
: 420 0541 1
: 421 0542 1 --
: 422 0543 1
: 423 0544 2 BEGIN
: 424 0545 2
: 425 0546 2 IF .GCA_NIA GTR 1 THEN : 1 or zero means all accounted for.
: 426 0547 2 ermn(rnfnic, .gca_nia - 1); : Subtract off reported negative indent
: 427 0548 2
: 428 0549 2 gca_nia = 0; : This starts the counting over again.
: 429 0550 1 END; : End of REMNEG

```

		0004 00000	.ENTRY REMNEG, Save R2	: 0521
	52 00000000G	EF 9E 00002	MOVAB GCA+160, R2	: 0546
	01	62 D1 00009	CMPL GCA+160, #1	
		11 15 0000C	BLEQ 1\$	
7E	62	01 C3 0000E	SUBL3 #1, GCA+160, -(SP)	: 0547
	00000000G	8F DD 00012	PUSHL #RNFNIC	
	00000000G	02 FB 00018	CALLS #2, ERMN	
	EF	62 D4 0001F	CLRL GCA+160	
		1\$: 04 00021	RET	: 0549
				: 0550

: Routine Size: 34 bytes, Routine Base: \$CODE\$ + 01CB

: 430 0551 1

```

: 432 0552 1 %SBTTL 'XMARG -- Controls R/L margin violation error messages'
: 433 0553 1 GLOBAL ROUTINE xmarg : NOVALUE =
: 434 0554 1
: 435 0555 1 !++
: 436 0556 1 FUNCTIONAL DESCRIPTION:
: 437 0557 1
: 438 0558 1 XMARG keeps track of attempted margin crossings (i.e., left
: 439 0559 1 margin exceeds right margin). To avoid issuing so many
: 440 0560 1 messages that the user would be annoyed, it only issues
: 441 0561 1 an error message for the first such attempt. Other violations
: 442 0562 1 are simply counted; the summary count is output by REMMRG later.
: 443 0563 1
: 444 0564 1 FORMAL PARAMETERS: None
: 445 0565 1
: 446 0566 1 IMPLICIT INPUTS: None
: 447 0567 1
: 448 0568 1 IMPLICIT OUTPUTS: None
: 449 0569 1
: 450 0570 1 ROUTINE VALUE:
: 451 0571 1 COMPLETION CODES: None
: 452 0572 1
: 453 0573 1 SIDE EFFECTS: None
: 454 0574 1
: 455 0575 1 --
: 456 0576 1
: 457 0577 2 BEGIN
: 458 0578 2
: 459 0579 2 IF .gca_xmarg EQL 0 THEN
: 460 0580 2   erml(rnfmrg);
: 461 0581 2
: 462 0582 2   gca_xmarg = .gca_xmarg + 1;
: 463 0583 1 END;                               !End of XMARG

```

52 00000000G	EF 9E 00000	0004 00000	.ENTRY XMARG, Save R2	: 0553
	62 D5 00009		MOVAB GCA+164, R2	: 0579
	0D 12 0000B		TSTL GCA+164	: 0580
00000000G EF	8F DD 0000D		BNEQ 1\$: 0582
	01 FB 00013		PUSHL #RNFMRG	: 0583
	62 D6 0001A	1\$:	CALLS #1, ERML	
	04 0001C		INCL GCA+164	
			RET	

: Routine Size: 29 bytes, Routine Base: \$CODE\$ + 01ED

: 464 0584 1

```

: 466 0585 1 %SBTTL 'REMMRG -- reports # of crossed margin errors'
: 467 0586 1 GLOBAL ROUTINE REMMRG : NOVALUE =
: 468 0587 1
: 469 0588 1 ++
: 470 0589 1 FUNCTIONAL DESCRIPTION:
: 471 0590 1
: 472 0591 1 Issues an error message indicating how many crossed
: 473 0592 1 margins have been detected. Then resets the count to zero.
: 474 0593 1
: 475 0594 1 Used together with XMARG.
: 476 0595 1
: 477 0596 1 FORMAL PARAMETERS: None
: 478 0597 1
: 479 0598 1 IMPLICIT INPUTS: None
: 480 0599 1
: 481 0600 1 IMPLICIT OUTPUTS: None
: 482 0601 1
: 483 0602 1 ROUTINE VALUE:
: 484 0603 1 COMPLETION CODES: None
: 485 0604 1
: 486 0605 1 SIDE EFFECTS: None
: 487 0606 1
: 488 0607 1 --
: 489 0608 1
: 490 0609 2 BEGIN
: 491 0610 2
: 492 0611 2 IF .gca_xmarg GTR 1 THEN ! 1 or zero means all accounted for.
: 493 0612 2 ermn(rnfmrc, .gca_xmarg - 1); ! Subtract off reported negative indent
: 494 0613 2
: 495 0614 2 gca_xmarg = 0; ! This starts the counting over again.
: 496 0615 1 END; ! End of REMMRG

```

		0004 00000	.ENTRY REMMRG, Save R2	: 0586
	52 00000000G	EF 9E 00002	MOVAB GCA+164, R2	: 0611
	01	62 D1 00009	CMPL GCA+164, #1	: 0612
		11 15 0000C	BLEQ 1\$: 0614
7E	62	01 C3 0000E	SUBL3 #1, GCA+164, -(SP)	: 0615
	00000000G	8F DD 00012	PUSHL #RNFMRC	
	00000000G EF	02 FB 00018	CALLS #2, ERMN	
		62 D4 0001F 1\$:	CLRL GCA+164	
		04 00021	RET	

: Routine Size: 34 bytes. Routine Base: \$CODES + 020A

```

: 497 0616 1
: 498 0617 1 END
: 499 0618 0 ELUDOM !End of module

```

RSDERM
V04-000

Misc. error checks and messages.
REMMRG -- reports # of crossed margin errors

J 12
16-Sep-1984 01:40:15
14-Sep-1984 13:07:56

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]RSDERM.BLI;1

Page 19
(12)

RTI
VO

PSECT SUMMARY

Name	Bytes	Attributes
\$CODE\$	556	NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
\$OWNS\$	4	NOVEC, WRT, RD , NOEXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
\$SPLIT\$	28	NOVEC,NOWRT, RD , NOEXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

Library Statistics

File	----- Symbols -----	Pages	Processing
	Total Loaded Percent	Mapped	Time
\$255\$DUA28:[SYSLIB]XPORT.L32:1	590 0 0	252	00:00.1
\$255\$DUA28:[RUNOFF.SRC]DSRLIB.L32:1	1248 28 2	86	00:00.3

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:RSDERM/OBJ=OBJ\$:RSDERM MSRC\$:RSDERM/UPDATE=(ENH\$:RSDERM)

Size: 556 code + 32 data bytes
Run Time: 00:10.7
Elapsed Time: 00:28.5
Lines/CPU Min: 3478
Lexemes/CPU-Min: 15303
Memory Used: 58 pages
Compilation Complete

0348 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RNODEF
LIS

RSDEMR
LIS

RTERM
LIS

RUNOFF
LIS

RSKIPS
LIS

SAURES
LIS

RNOUWS
LIS

RNOUWS
LIS

RNODAT
LIS

RNFERM
LIS